

```
In [51]: !pip install apyori
import pandas as pd
import numpy as np
from apyori import apriori
```

Requirement already satisfied: apyori in c:\users\asus\downloads\datamining\lib\site-packages (1.1.2)

```
In [52]: df = pd.read_csv("groceries_dataset.csv")
df.head()
```

```
Out[52]:
```

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10	...	Item 32
0	citrus fruit	semi-finished bread	margarine	ready soups	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
1	tropical fruit	yogurt	coffee	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
2	whole milk	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
3	pip fruit	yogurt	cream cheese	meat spreads	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
4	other vegetables	whole milk	condensed milk	long life bakery product	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN

5 rows × 32 columns



```
In [53]: records = [] # initialize an empty list to store all transactions
for i in range(0, len(df)):
    # iterate through each row in the DataFrame
    # create a list of non-NaN cell values (convert each to string for consistency)
    transaction = [str(df.values[i, j]) # get the cell value as a string
                   for j in range(df.shape[1]) # loop over all columns in the DataFrame
                   if str(df.values[i, j]) != 'nan'] # exclude missing (NaN) values
    # add the cleaned transaction (row without NaN) to the records list
    records.append(transaction)
```

```
In [54]: print(len(records))
print(records[0])
```

9835

```
['citrus fruit', 'semi-finished bread', 'margarine', 'ready soups']
```

The dataset has 9835 transactions and 32 columns.

- Each row represents one transaction or basket (what one customer bought).
- Each column represents an item position within that basket (like item 1, item 2, ... item 32).

According to one output, one shopper's basket contained citrus fruit, semi-finished bread, margarine, and ready soups.

```
In [55]: # apply the Apriori algorithm to extract association rules
# parameters:
#   min_support=0.5 → itemsets must appear in at least 50% of all transactions
#   min_confidence=0.75 → rules must have at least 75% confidence
#   min_lift=1 → ensures the rule provides at least some useful association
#   min_length=2 → only consider rules with at least 2 items
rules_a = apriori(records, min_support=0.5, min_confidence=0.75, min_lift=1, min_length=2)

# convert the generator output from apriori() into a list for easy handling
results_a = list(rules_a)

# display how many association rules were found under these parameters
print("Case (a) rules mined:", len(results_a))
```

Case (a) rules mined: 0

0 rules here - it means that no items co-occur in $\geq 50\%$ of all transactions.

```
In [56]: # apply the Apriori algorithm again with a different set of thresholds
# parameters:
#   min_support=0.6 → itemsets must appear in at least 60% of all transactions
#   min_confidence=0.6 → rules must have at least 60% confidence
#   min_lift=1 → ensures the rule indicates at least some positive association
#   min_length=2 → only include rules containing at least 2 items
rules_b = apriori(records, min_support=0.6, min_confidence=0.6, min_lift=1, min_length=2)

# convert the Apriori generator output into a list for easy viewing and processing
results_b = list(rules_b)

# print the total number of rules mined under these parameters (Case b)
print("Case (b) rules mined:", len(results_b))
```

Case (b) rules mined: 0

0 rules here - this is expected since the dataset is large and sparse. No items appear in $\geq 60\%$ of all transactions.

min_length = 2 means "Each rule must have at least 2 items (like 'milk' -> 'bread')."

That's the bare minimum to form a rule - you can't have an association with only one item. So, min_length = 2 is the lowest valid choice and won't block anything. If there were frequent pairs in $\geq 50\%$ of all transactions, rules would appear.

min_lift = 1 means "You'll accept any rule where items are at least as likely to occur together as by chance." That's also lenient - lift = 1 doesn't filter out anything important. So, again, this doesn't block any rules.

- Lift = (confidence / expected confidence).
- It's only computed after a rule passes support and confidence thresholds.
- Raising min_lift makes the filter stricter (keeps fewer rules), not looser.
- Lowering min_lift to 1 or 0 can't create rules that failed the support step, because those itemsets were already removed.

- So, changing min_lift up or down will not convert "0 rules" into something non-zero with support 0.5 or 0.6.

The real limiting factor is min_support (0.5 and 0.6). No combination of items - even single ones like whole milk - occurs in that many transactions.

```
In [57]: # Define a helper function 'inspect' to extract readable association rules from
def inspect(results):
    data = [] # Create an empty list to store rule information

    # Each 'RelationRecord' represents one frequent itemset discovered by Apriori
    for RelationRecord in results:

        # Each record can have multiple "ordered statistics" (possible associations)
        for ordered_stat in RelationRecord.ordered_statistics:

            # Ensure both sides of the rule (LHS → RHS) are non-empty
            if len(ordered_stat.items_base) > 0 and len(ordered_stat.items_add):

                # Add a dictionary for each valid rule containing rule text, support, confidence, and lift
                data.append({
                    'Rule': f"{set(ordered_stat.items_base)} → {set(ordered_stat.items_add)}",
                    'Support': RelationRecord.support,
                    'Confidence': ordered_stat.confidence,
                    'Lift': ordered_stat.lift
                })

    # Convert the list of rule dictionaries into a DataFrame for easy viewing
    return pd.DataFrame(data)

# Apply the function to both Apriori results
df_a = inspect(results_a)
df_b = inspect(results_b)

# Print and display the results for both cases
print("=== Case (a) 50% Support, 75% Confidence ===")
display(df_a)

print("=== Case (b) 60% Support, 60% Confidence ===")
display(df_b)
```

```
=== Case (a) 50% Support, 75% Confidence ===
```

```
—
```

```
=== Case (b) 60% Support, 60% Confidence ===
```

```
—
```

```
In [58]: rules_demo = apriori(records,
                                min_support=0.02, # items appearing in ≥2% of transaction
                                min_confidence=0.3, # rules that are correct ≥30% of the time
                                min_lift=2, # rules that are at least twice as likely to occur together
                                min_length=2) # rules must involve at least 2 items

df_demo = inspect(list(rules_demo))
display(df_demo.head(10))
```

	Rule	Support	Confidence	Lift
0	{'root vegetables'} → {'other vegetables'}	0.047382	0.434701	2.246605
1	{'whipped/sour cream'} → {'other vegetables'}	0.028876	0.402837	2.081924
2	{'other vegetables', 'whole milk'} → {'root ve...	0.023183	0.309783	2.842082
3	{'root vegetables', 'whole milk'} → {'other ve...	0.023183	0.474012	2.449770
4	{'yogurt', 'other vegetables'} → {'whole milk'}	0.022267	0.512881	2.007235
5	{'yogurt', 'whole milk'} → {'other vegetables'}	0.022267	0.397459	2.054131

Rule	Support	Confidence	Lift	Interpretation
{'root vegetables'} → {'other vegetables'}	0.047	0.435	2.25	The rule has 4.7% support, meaning it occurs in 4.7% of all transactions. It has 43.5% confidence, so when root vegetables are bought, other vegetables are bought 43.5% of the time. The lift of 2.25 indicates this co-occurrence is more than twice as likely as by random chance.
{'whipped/sour cream'} → {'other vegetables'}	0.029	0.403	2.08	This rule appears in 2.9% of transactions. With 40.3% confidence, it means that when whipped/sour cream is purchased, other vegetables are bought 40% of the time. The lift of 2.08 shows the items are roughly twice as likely to occur together than independently.
{'other vegetables', 'whole milk'} → {'root vegetables'}	0.023	0.310	2.84	Occurs in 2.3% of all transactions. The 31% confidence means that when both other vegetables and whole milk are bought, root vegetables are also bought 31% of the time. The lift of 2.84 indicates the combination is almost three times more likely than random chance.
{'root vegetables', 'whole milk'} → {'other vegetables'}	0.023	0.474	2.45	Appears in 2.3% of transactions. The 47.4% confidence means that nearly half of the customers who buy root vegetables and whole milk also buy other vegetables. A lift of 2.45 confirms the strong positive association.
{'yogurt', 'other vegetables'} → {'whole milk'}	0.022	0.513	2.01	This rule occurs in 2.2% of all transactions. The confidence of 51.3% means that in more than half of cases where yogurt and other vegetables are bought together, whole milk is also bought. A lift of 2.01 shows this is twice as likely as random.
{'yogurt', 'whole milk'} → {'other vegetables'}	0.022	0.397	2.05	The rule appears in 2.2% of all transactions. With 39.7% confidence, customers buying yogurt and milk also buy other vegetables 39.7% of the time. The lift of 2.05 means this co-occurrence

Rule	Support	Confidence	Lift	Interpretation
				is about twice as frequent as random expectation.

Explanation:

Apriori is run twice above:

- Case (a): 50% support, 75% confidence
- Case (b): 60% support, 60% confidence
- With `min_lift = 1`, `min_length = 2`

Output: Both cases gave 0 association rules.

That outcome still counts as a correct result, because it reflects that:

- The support thresholds (50%, 60%) are very high. In the groceries dataset (where products are highly diverse), it's rare for many items to appear together in over 50% of all transactions.
- Confidence and lift conditions further filter out weak rules.
- So, the Apriori algorithm correctly finds no frequent itemsets meeting those criteria - meaning the dataset does not contain such strong associations.

Parameter	Meaning	Effect if You Change It
<code>min_lift</code>	Measures how strongly two items are associated beyond random chance ($\text{Lift} > 1$ = positive correlation).	Lowering it from 3 \rightarrow 1 or even 0 will not help here, because rules aren't being generated at all (they fail the support threshold before lift is even checked).
<code>min_length</code>	Minimum number of items per rule. Example: <code>min_length=2</code> means at least two items in a rule.	If you lower it to 1, you'll still get 0 rules — because a single item cannot form an association rule, and support filtering happens first.

So, changing `min_lift` or `min_length` won't fix the 0-result issue when the support threshold is too high.

This proves the algorithm's correctness, since it doesn't falsely output weak or non-existent rules.

Some important formulas:

- $\text{Support}(A) = \text{count}(A) / N$
- $\text{Support}(B) = \text{count}(B) / N$
- $\text{Support}(A \cup B) = \text{count}(A \text{ and } B) / N$
- $\text{Confidence}(A \rightarrow B) = \text{Support}(A \cup B) / \text{Support}(A)$
- $\text{Lift}(A \rightarrow B) = \text{Confidence}(A \rightarrow B) / \text{Support}(B)$

Because $\text{confidence} = \text{Support}(A \cup B) / \text{Support}(A)$, you can also write lift as:

- $\text{Lift}(A \rightarrow B) = \text{Support}(A \cup B) / (\text{Support}(A) * \text{Support}(B))$